22i-1552_SyedArhamAhmed_CY-B

1. **What is the difference between the directives, ‘=’, ‘equ’, and ‘textequ’?**

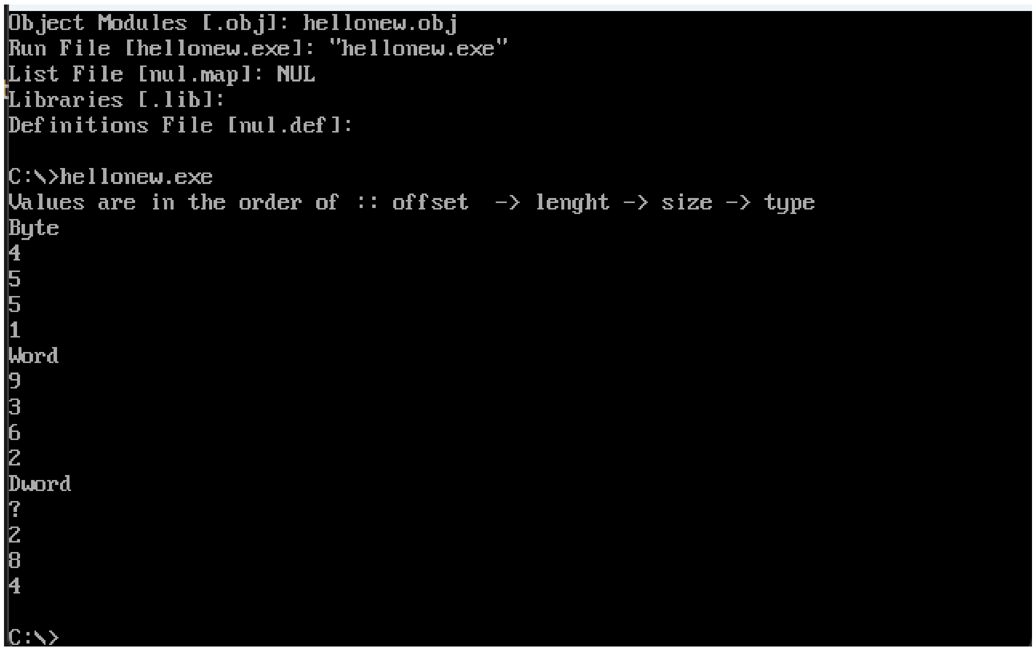
   EQU is used for assigning integer values to variables such as:
   Salary EQU 300h
   Savings EQU Salary
   Along with this any symbol that is assigned integer values using EQU cannot be redefined in the same source file.

   TEXTEQU is similar to EQU, it creates text macro such as:
   Name TEXTEQU “Arham 22i-1552”
   Also, a symbol defined by TEXTEQU can be re-defined at any time in the source file.

2. **Verify little endian order of saving variables in memory.**

   In little Indian ordering the least significant byte of a multi-byte variable is stored at the lowest memory address while the most significant byte is stored at the highest memory address.
   For e.g.
   Num word 101112h will be stored as:
   12 11 10 00 in memory.

Q.3.



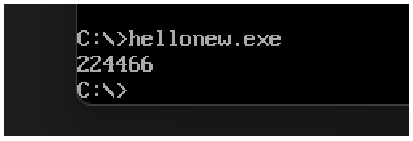
```
Object Modules [.obj]: hellonew.obj
Run File [hellonew.exe]: "hellonew.exe"
List File [nul.map]: NUL
Libraries [.lib]:
Definitions File [nul.def]:

C:\>hellonew.exe
Values are in the order of :: offset  -> lenght -> size -> type
Byte
4
5
5
1
Word
9
3
6
2
Dword
?
2
8
4

C:\>
```

```
dosseg
.model small
.stack 100h

.data
```

```
	byte_array db 'a','b','c','d','e'
	word_array dw 1234h,5678h,9ABCh
	dword_array dd 12345678h, 87654321h
    string db "Byte",'$'
    string_2 db "Word",'$'
    string_3 db "Dword",'$'
    string_4 db "Values are in the order of :: offset  -> lenght -> size ->
type",'$'
    array_length db ?, ?, ?
    array_size db ?, ?, ?
.code

main proc

	mov ax, @data
    mov ds, ax

    mov dx,offset string_4
    mov ah,9
    int 21h

    mov ah, 0Eh       ;print new line sequence
    mov al, 0Dh
    int 10h
    mov al, 0Ah
    int 10h

    mov dx,offset string  ; prints byte
    mov ah,9
    int 21h

    mov ah, 0Eh       ;print new line sequence
    mov al, 0Dh
    int 10h
    mov al, 0Ah
    int 10h

    mov dx,offset byte_array
    add dx,48
    mov ah,2
    int 21h

    mov dl, 0Ah
    mov ah, 2h
    int 21h

    mov dl, byte ptr lengthof byte_array
    add dl, 48
    mov ah, 2
    int 21h

    mov dl, 0Ah
    mov ah, 2h
    int 21h

    mov dl, byte ptr sizeof byte_array
    add dl, 48
    mov ah, 2
    int 21h

    mov dl, 0Ah
    mov ah, 2h
    int 21h

    mov dl,byte ptr type byte_array
    add dl,48
    mov ah,2
    int 21h

    mov dl, 0Ah
    mov ah, 2h
    int 21h


    ; yahan sy word ka hy

    mov dx,offset string_2  ; prints word
    mov ah,9
    int 21h

    mov ah, 0Eh       ;print new line sequence
    mov al, 0Dh
    int 10h
    mov al, 0Ah
    int 10h

    mov dx,offset word_array
    add dx,48
    mov ah,2
    int 21h

    mov dl, 0Ah
    mov ah, 2h
    int 21h

    mov dl, byte ptr lengthof word_array
    add dl, 48
    mov ah, 2
    int 21h

    mov dl, 0Ah
    mov ah, 2h
    int 21h

    mov dl, byte ptr sizeof word_array
    add dl, 48
    mov ah, 2
    int 21h

    mov dl, 0Ah
    mov ah, 2h
    int 21h

    mov dl,byte ptr type word_array
    add dl,48
    mov ah,2
    int 21h

    mov dl, 0Ah
    mov ah, 2h
    int 21h


    ; yahan sy dword ka hy

    mov dx,offset string_3  ; prints dword
    mov ah,9
    int 21h

    mov ah, 0Eh       ;print new line sequence
    mov al, 0Dh
    int 10h
    mov al, 0Ah
    int 10h

    mov dx,offset dword_array
    add dx,48
    mov ah,2
    int 21h

    mov dl, 0Ah
    mov ah, 2h
    int 21h

    mov dl, byte ptr lengthof dword_array
    add dl, 48
    mov ah, 2
    int 21h

    mov dl, 0Ah
    mov ah, 2h
    int 21h

    mov dl, byte ptr sizeof dword_array
    add dl, 48
    mov ah, 2
    int 21h

    mov dl, 0Ah
    mov ah, 2h
    int 21h

    mov dl,byte ptr type dword_array
    add dl,48
    mov ah,2
    int 21h

    mov dl, 0Ah
    mov ah, 2h
    int 21h


    mov ah,4ch
    int 21h
```

```
main endp
end main
```

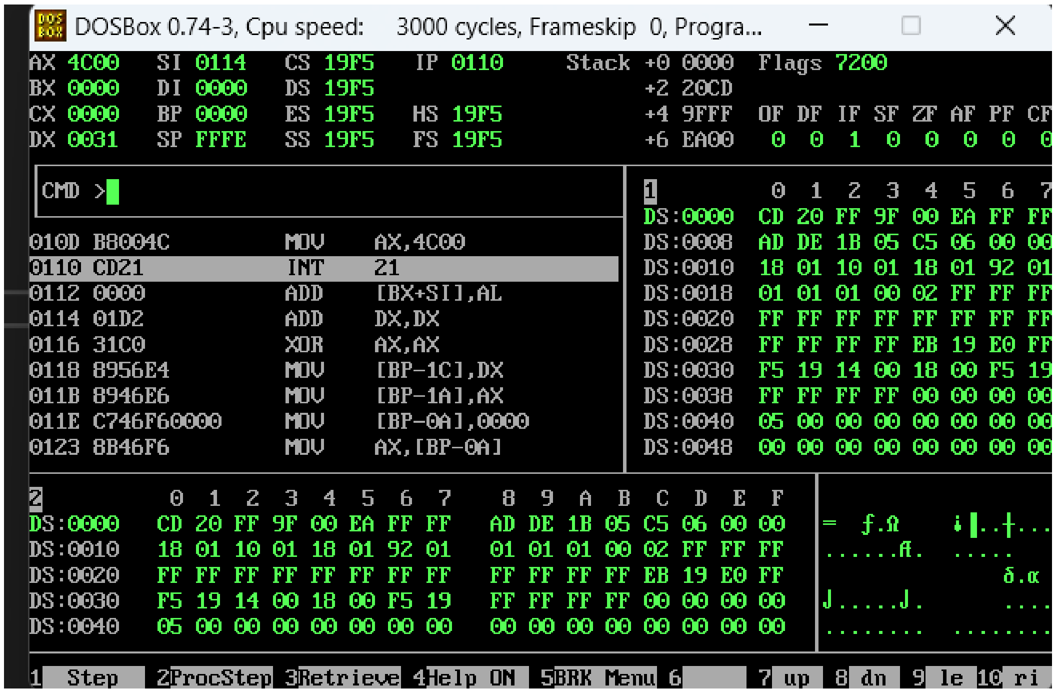
Q.4.



```
C:\>hellonew.exe
224466
C:\>
```

```
dosseg
.model small
.stack 100h

.data
    byte_array db 1,2,3,4,5,6
    rollnumber db "1552",'$'

    array_length db ?, ?, ?
    array_size db ?, ?, ?
.code

main proc
    mov ax, @data
    mov ds, ax

    mov al, rollnumber + 3
    sub al, '0'
    and al, 1

    cmp al, 0

    je add_even
    jmp add_odd

add_even:
    mov si, offset byte_array
    mov cx, lengthof byte_array
    mov bl, 1

add_loop:
    mov al, [si]
    add al, bl
    mov [si], al
    add si, 2
    loop add_loop

    jmp print_array

add_odd:
    mov si, offset byte_array + 1
    mov cx, lengthof byte_array - 1
    mov bl, 2

add_loop2:
    mov al, [si]
    add al, bl
    mov [si], al
    add si, 2
    loop add_loop2

print_array:
    mov si, offset byte_array
    mov cx, lengthof byte_array

print_loop:
    mov dl, [si]
    add dl, '0'
    mov ah, 02h
    int 21h
    inc si
    loop print_loop


    mov ah, 4Ch
    int 21h

main endp
end main
```

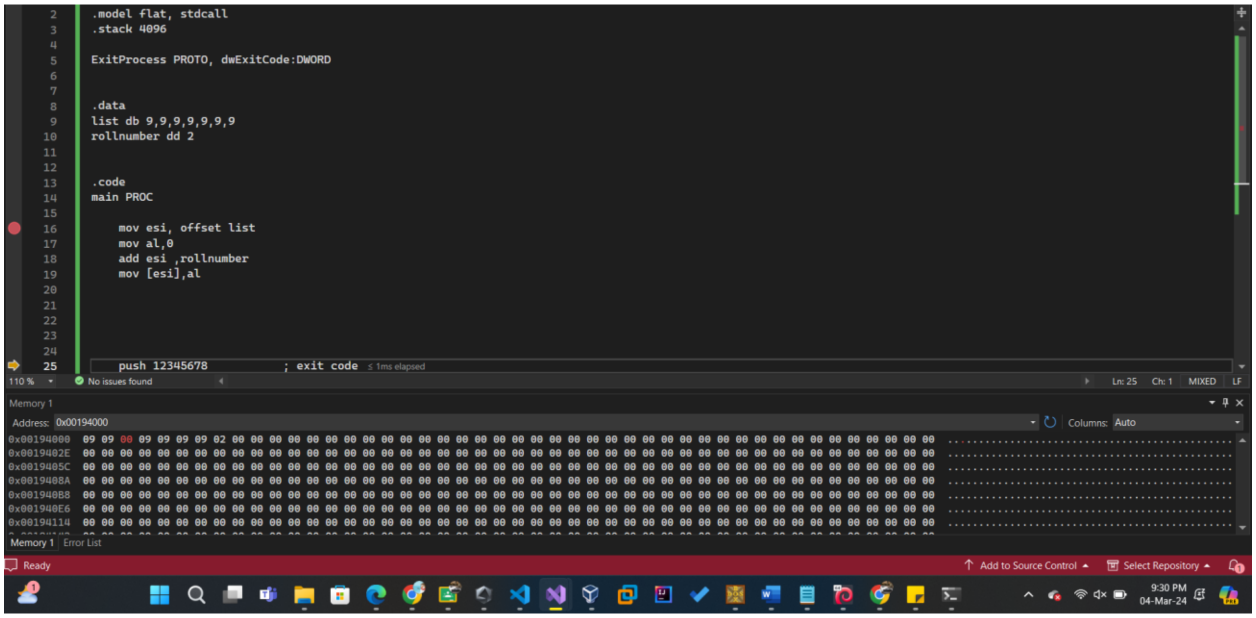
Q.5.


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
AX 4C00    SI 0114    CS 19F5    IP 0110      Stack +0 0000  Flags 7200
BX 0000    DI 0000    DS 19F5                       +2 20CD
CX 0000    BP 0000    ES 19F5    HS 19F5            +4 9FFF  OF DF IF SF ZF AF PF CF
DX 0031    SP FFFE    SS 19F5    FS 19F5            +6 EA00   0  0  1  0  0  0  0  0

 CMD >                                       1          0  1  2  3  4  5  6  7
                                             DS:0000  CD 20 FF 9F 00 EA FF FF
010D B8004C        MOV     AX,4C00           DS:0008  AD DE 1B 05 C5 06 00 00
0110 CD21          INT     21                DS:0010  18 01 10 01 18 01 92 01
0112 0000          ADD     [BX+SI],AL        DS:0018  01 01 01 00 02 FF FF FF
0114 01D2          ADD     DX,DX             DS:0020  FF FF FF FF FF FF FF FF
0116 31C0          XOR     AX,AX             DS:0028  FF FF FF FF EB 19 E0 FF
0118 8956E4        MOV     [BP-1C],DX        DS:0030  F5 19 14 00 18 00 F5 19
011B 8946E6        MOV     [BP-1A],AX        DS:0038  FF FF FF FF 00 00 00 00
011E C746F60000    MOV     [BP-0A],0000      DS:0040  05 00 00 00 00 00 00 00
0123 8B46F6        MOV     AX,[BP-0A]        DS:0048  00 00 00 00 00 00 00 00

2          0  1  2  3  4  5  6  7    8  9  A  B  C  D  E  F
DS:0000   CD 20 FF 9F 00 EA FF FF   AD DE 1B 05 C5 06 00 00   =  ƒ.Ω    ¡▌..┼...
DS:0010   18 01 10 01 18 01 92 01   01 01 01 00 02 FF FF FF   ......Æ.  .....
DS:0020   FF FF FF FF FF FF FF FF   FF FF FF FF EB 19 E0 FF             δ.α
DS:0030   F5 19 14 00 18 00 F5 19   FF FF FF FF 00 00 00 00   J.....J.      ....
DS:0040   05 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........  ........

1 Step  2ProcStep 3Retrieve 4Help ON  5BRK Menu 6        7 up  8 dn  9 le 10 ri
```

```
[org 0x100]

section .data
    rollnumber db 1

section .text
    mov si, rollnumber
    mov al, [si]
    add al, '0'
    mov dl, al

    mov ah, 0x02
    int 0x21

    mov ax, 0x4c00
    int 0x21
```

Q.6.



```
.model flat, stdcall
.stack 4096

ExitProcess PROTO, dwExitCode:DWORD


.data
list db 9,9,9,9,9,9,9
rollnumber dd 2


.code
main PROC

    mov esi, offset list
    mov al,0
    add esi ,rollnumber
    mov [esi],al






    push 12345678          ; exit code
```

```
Address: 0x00194000
0x00194000  09 09 00 09 09 09 09 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..............................................
0x0019402E  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..............................................
0x0019405C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..............................................
0x0019408A  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..............................................
0x001940B8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..............................................
0x001940E6  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..............................................
0x00194114  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..............................................
```

```
.386
.model flat, stdcall
.stack 4096

ExitProcess PROTO, dwExitCode:DWORD


.data
var dd 1552


.code
main PROC

    mov esi, offset list
       mov al,0
       add esi ,rollnumber
       mov [esi],al
```

```
      push 12345678           ; exit code
      call ExitProcess        ; Or call ExitProcess@4
main ENDP

      END main        ;specifying the program's entry point
```